Lecture 15: Nested Lists

CS 51P

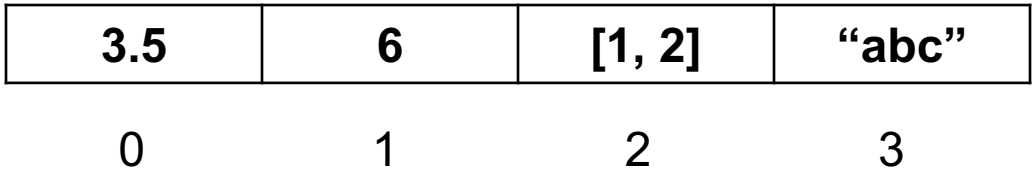
November 1, 2023

Access elements in an inner list

- A list that consists of an inner list

```
a_list = [3.5, 6, [1, 2], "abc"]
```

- a_list

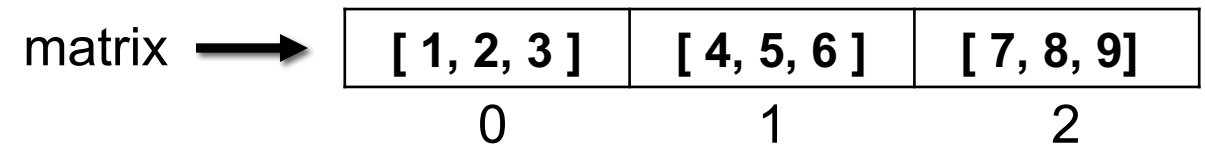


- a_list[2] is [1, 2]
  - a_list[2][0] is 1
  - a_list[2][1] is 2

- To access or modify elements, specify index in “outer” list first, then index in “inner” list

Nested lists

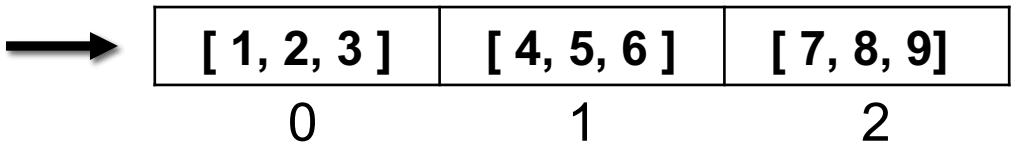
- Can create a **list of lists** aka a **nested list**!
- 2-D list is a list of lists
  - Each element of “outer” list is just another list (the inner list)
  - Can think of this as a matrix if inner lists have the same size
- Example:
  - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]



- matrix[0] → [1, 2, 3]
- matrix[1] → [4, 5, 6]
- matrix[2] → [7, 8, 9]

Nested lists

- Can create a **list of lists** aka a **nested list**!
- 2-D list is a list of lists
  - Each element of “outer” list is just another list (the inner list)
  - Can think of this as a matrix if inner lists have the same size
- Example:
  - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]



- matrix[0][0] → 1
- matrix[1][0] → 4
- matrix[2][2] → 9

Example

- Nested lists can be N-dimensional
- Inner lists do not have to be in the same size

```
a_list = [ [4, [True, False], 6, 8], [888, 999] ]

if alist[0][1][0]:
    print(alist[1][0])
else:
    print(alist[1][1])
```

Example

- Define a function nested_total that takes a list of lists of ints and returns the sum of all the values.

```
list = [[1,2], [3], [4,5,6]]
sum = nested_total(list)
print(sum)
```

```
21
```

Exercise

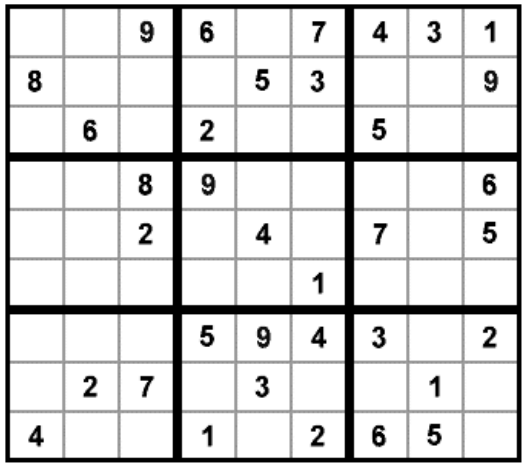
- Define a function nested_avg that takes a list of lists of ints and returns a list with each sublist averaged

```
list = [[1,2], [3], [4,5,6]]
list_avg = nested_avg(list)
print(list_avg)
```

```
[1.5, 3.0, 5.0]
```

Example - Sudoku

LEVEL: Beginner



		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3			1	
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],
         [8,0,0,0,5,3,0,0,9],
         [0,6,0,2,0,0,5,0,0],
                  ...
         [4,0,0,1,0,2,6,5,0]]
```

- Rules of the game:
  - Grid of 9x9 spaces
  - Each row, column, and 3x3 square needs to have the numbers 1-9, without repeating any numbers within row, column or square

Example

LEVEL: Beginner

		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3			1	
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],
         [8,0,0,0,5,3,0,0,9],
         [0,6,0,2,0,0,5,0,0],
                   ...
         [4,0,0,1,0,2,6,5,0]]
```

- write a function `set_value` that takes a nested list board and ints i, j, n and updates the (i,j)th entry of board to be the value n
- write a function `check_row` that takes an int i and a nested list board. The function should return True if and only if row i contains each integer from 1 through 9 exactly once.

Exercise

LEVEL: Beginner

		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3			1	
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],
         [8,0,0,0,5,3,0,0,9],
         [0,6,0,2,0,0,5,0,0],
                    ...
         [4,0,0,1,0,2,6,5,0]]
```

- write a function `check_column` that takes an int j and a nested list board. The function should return True if and only if column i contains each integer from 1 through 9 exactly once.
- write a function `check_block` that takes ints i and j and a nested list board. The function should return True if and only if the 3x3 block starting at row i, column j contains each integer from 1 through 9 exactly once
- write a function `check_solution` that takes a nested list board and returns True if and only if board represents a correctly solved puzzle.